COMP90051 Statistical Machine Learning, Semester 2, 2019
Project 1: Who Tweeted That?

Group - 134
Swapnil Shailee - sshailee (952247)
Shreyas Walvekar - swalvekar (961621)
Nitish Mathur - nitishm (954892)

I. INTRODUCTION

The report endeavours to elucidate the various facets of the feature engineering methods and machine learning models implemented as the part of assignment COMP90051 as specified in the assignment specification [1]. In doing so, it provides an overview of the text corpus (including training and testing data), the strategies implemented as a part of feature engineering and a brief overview of various algorithms used to produce the inferences. Additionally, the report documents a comparison of performance of the various models used while providing an insight on the choice of parameter configuration. Finally, the report delves into the knowledge gained in pertinence to derive a model that yields best result by minimizing the problem of underfitting and overfitting.

II. DATASET OVERVIEW

The twitter corpus is a collection of raw tweets divided into training and test data. The training data consists of 3,28,932 tweets from 9,297 individual users and each tweet is tab-delimited with a user ID. The test data contains 35,437 unlabelled tweets [1]. Many of the raw tweets contains URL links, various punctuations and a very informal style of writing which has created a diverse vocabulary containing more than 0.1 million words.

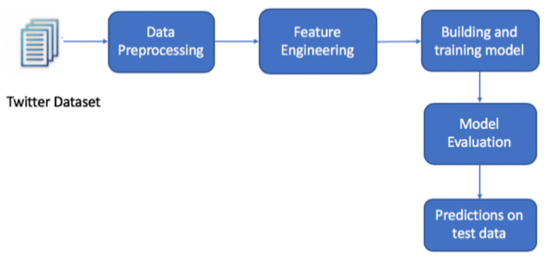

Figure 1 System Architecture

III. FEATURE ENGINEERING

Feature engineering is the method of drawing out relevant features from the pre-processed data that will be useful in building and training the machine learning model.
In the data pre-processing step, the training dataset which is a text file of raw tweets has been first filtered out by removing http links, re-tweet symbols, punctuations and twitter handles. The data is then stored in a structured tabular format using *pandas* python package [2].
Different feature engineering approaches like count vectors as features also known as the bag of words model and TF-IDF model with different n-gram variations have been considered. The bag of words model defines the occurrence of all words in the complete dataset. TF-IDF represents the importance of a words in a dataset which is based on the vector space model and is widely used in the industry.

Features	Vectorizer
Bag of words model (Count Vectors as features)	Count Vectorizer feature extraction has been used which stores the token count of the entire dataset in a matrix format [3].
TF-IDF vectors as features (Word level)	TF-IDF Vectorizer as feature extraction where the word is set as analyser [**4**]. TF-IDF Vectorizer as feature extraction with sublinear term frequency and words with more than 50% occurrence in the dataset are removed [4].
TF-IDF vectors as features (N-gram level)	TF-IDF Vectorizer as feature extraction where word is set as analyser and combination of unigram and bigram scores of words are calculated [**4**]. TF-IDF Vectorizer as feature extraction where word is set as analyser and the score is calculated using various n-gram range [4].

Table 1 Feature Engineering (Features and Vectorizer)

IV. MACHINE LEARNING ALGORITHMS

i. K Nearest Neighbours

Nearest neighbour classifiers simply store instances of the training data and classification is computed from a simple majority vote of nearest neighbours. The nearest neighbours are neighbours of a query point that represents the test string in question. The k-Nearest Neighbours algorithm defines a parameter k for the number of neighbours to consider for voting. Voting here refers to picking out the class that occurs the greatest number of times among the k neighbours.

Training phase of the algorithm consists of storing the examples in a multidimensional feature space along with their associated class label. Later, during classification, the neighbours are chosen with respect to the query point based on distance. Commonly for continuous variables, Euclidean distance is used, whereas Hamming distance is more suited for text classification. Sklearn package provides another metric, the Minkowski distance, which is a generalization of Euclidean and Manhattan distances.
Lastly, the neighbours can be assigned weights depending inversely on their distance, so that nearer points have a higher influence in deciding the labels. In a massively multiclass environment, KNN can become biased towards labels that occur sparsely yet frequently across the entire vector space.

ii. Perceptron

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function that maps from an n-dimensional input to an m dimensional output. In out project, the target output is a single label, so it learns in a non-linear approximation for classification. It is a neural network which can have multiple hidden layers between input and output. The first layer takes the set of features as the input and transforms into the output of the next layer as a weighted linear summation, this in turn passes through a non-linear transformation with a hyperbolic tan function. This model can do a real time fit of data, in an online environment.
The classification algorithm trains using gradient descent, where the gradients are calculated using backpropagation. Final classification is done by finding probabilities of labels by minimizing their cross-entropy loss function.
In the multi-label classification, meaning, that when a data can lie in multiple labels, this feature allows choosing of a threshold that can be used to pick out the top labels [6].

iii. Support Vector Machines

An SVM classifier solves the optimization problem of finding an n-dimensional hyperplane that maximizes the separation of n+1-dimensional feature space. It achieves multiclass classification by finding the optimization for all k classes and comparing those amongst each other. Sklearn provides SVC and LinearSVC models.
SVC supports various kernels like RBF, Polynomial and Sigmoid for fitting the data. For classification this kind uses a 'one-against-one' approach as a n*(n-1)/2 number of classifiers are constructed which train data from two classes each.
LinearSVC on the other hand has a linear kernel and a 'one-versus-rest' multiclass strategy where each class has its own optimal solution computed for the hyperplane separation problem and they are intercepted to find a point of classification that assigns the final label [7].

V. MODEL SELECTION AND ANALYSIS

This section describes the combination of feature engineering techniques and algorithms used as mentioned in section III and IV respectively as part of model selection and compares the performance of each model in terms of training time, prediction time and accuracy obtained.
In each of the experiment, the training corpus was divided into training and validation set with a ratio of (9:1) giving 2,96,039 tweets used for training and 32,893 tweets used for evaluation. All the algorithms and text vectorizers used were part of sci-kit library.

i. K-Nearest Neighbour

KNeighborsClassifier was trained with the raw dataset of tweets by converting the vectors using Count Vectorizer having the following parameters:

a. *metric 'euclidean', n_neighbors* 5 – This parametric tuning allowed the model to train very quickly but the model was extremely underfitting hence resulting in very low accuracy. The probable reason for this behaviour was the high sensitivity of the algorithm towards irrelevant features
b. *metric 'hamming, n_neighbors 5* – This model is computationally expensive to build probably because of the very high number of features in the dataset (133489).

ii. Perceptron

MLPClassifier is more suitable for large datasets, and it was trained with vectors created by TFIDF Vectorizer. The time required to train the model depends on the number of features and the following results were obtained with different feature settings:

a. *max_features '133489'* – This model took a longer time to train but was able to produce better inferences. This probably could have happened because the classifier adjusts the whole model every time it makes a mistake.
b. *max_features '20000'* – This model was comparatively faster to train but had a poor accuracy because the word-vector index of maximum features was less than 0.1 and by limiting the features to 20000 most of the crucial features were removed which severely impacted the accuracy.

The common behaviour obtained from both the configurational setup was the time to produce an inference by a Perceptron classifier was very high.

iii. Support Vector Machine

LinearSVC and SVC classifiers were trained on the vectors created by TFIDF Vectorizer. The loss function used for both classifiers was 'l2'. The model was developed using 'unigram' parameters. With different hyperparameter tuning, following results were observed:

a. *analyzer 'char', kernel 'linear'* – With this configuration the number of features were very less (137) hence causing the model to underfit.

b. *analyzer 'char', kernel 'rbf'* – Model with this configuration was computationally expensive to build because with high number of features, rbf kernel requires higher number of iterations to converge.

c. *analyzer 'word', kernel 'linear'* – This model took a longer time to train when compared with KNN and Perceptron, but it outperformed every other model by giving higher accuracy and less prediction time. The main reason is because the features can be distinguished by linear hyperplanes hence minimizing the effect of overfitting.

Model (config)	Training Time	Prediction Time	Accuracy (%)
KNN (a)	3min 12sec	3min	25
KNN (b)	CE[1]	-	-
Perceptron (a)	43min	3hr	38.7
Perceptron (b)	1hr 10min	4.5 hr	52.1
SVM (a)	1hr 53 min	3.5 min	75.3
SVM (b)	2hr 2 min	3.5 min	62.4
SVM (c)	CE[1]	-	-

Table 2: Model Evaluation (Selection Parameters)

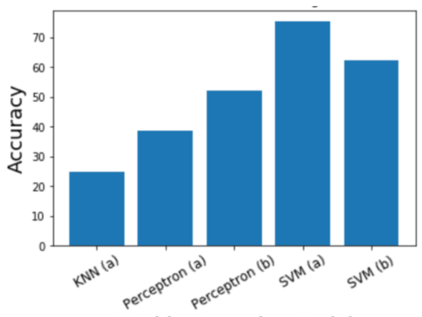

Figure 2: Model selection accuracy.

VI. CONCLUSION

The author attribution is a combination of data processing, feature engineering, building machine learning model, training model, model evaluation and predictions. Based on the model evaluation and critical analysis, KNN becomes progressively slower as the number of comparisons increases because of the large number of output labels as this task is based on extreme classification. KNN was sensitive to irrelevant features and hence it was underfitting the data. SVM machine learning model outperforms all the models implemented. It has shown better results because the TF-IDF features of users were linearly separable and hence it gave better results than RBF kernel. In future directions, the use of NLP features and discourse structure as features can improve the accuracy of the machine learning models.

[1] CE – Computationally Expensive (Could not train)

VII. REFERENCES

[1]"Ben Rubinstein "Statistical Machine Learning Project1: Who Tweeted That?". Available:https://app.lms.unimelb.edu.au/bbcswebdav/pid-7609732-dt-content-rid-65357770_2/courses/COMP90051_2019_SM2/project1/Project1.pdf

[2]*Cs224d.stanford.edu*, 2019. [Online]. Available:https://cs224d.stanford.edu/reports/RhodesDylan.pdf.

[3]"sklearn.feature_extraction.text.CountVectorizer — scikit-learn 0.21.3 documentation", *Scikit-learn.org*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html.

[4] "sklearn.feature_extraction.text.TfidfVectorizer — Feature Extraction from text", *Scikit-learn.org*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text

[5] "sklearn.neighbors.KNeighborsClassifier — scikit-learn 0.21.3 ...". [Online]. Available: http://scikitlearn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[6]"sklearn.neural_network.MLPClassifier — scikit-learn 0.21.3 ...". [Online]. Available: http://scikitlearn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[7]"1.4. Support Vector Machines — scikit-learn 0.21.3 documentation". [Online]. Available: http://scikit-learn.org/stable/modules/svm.html.

[8]"sklearn.svm.SVC — scikit-learn 0.21.3 documentation", *Scikit-learn.org*, 2019. [Online]. Available:https://scikitlearn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC.